HarvardX Data Science program capstone project

Vladimir Pedchenko

10/6/2021

Contents

1 Introduction

This is a report of HarvardX Data Science program capstone project (PH125.9x).

Goal of the project is to create a movie recommendation system using the MovieLens dataset. Recommender Systems (RSs) are software tools and techniques providing suggestions for items to be of use to a user[1]. It can be used in streaming services such YouTube or Netflix or in web-shops, to suggest user items which he/she, most likely, would buy.

In this specific case, we will try to predict rating of specific movie by specific user.

Before model building a model we have to perform Exploratory data analysis (EDA) and select metric for model estimation. Metric is defined by project goal definition: we have to reach root mean squared error (RMSE) < 0.86490. Thus, RMSE is our metric for this project.

Final RMSE estimation will be performed on the final hold-out validation test set, which we will not use for any other purposes, neither for training model nor for model selection.

2 Data preparation

Code bellow was provided by HarvardX. It downloads data and split it to two datasets: **edx** and **validation**. **Validation** data set will not be used in the code until the final validation of our selected and trained model. Data analysis, model selection/training will be performed on **edx** dataset

This chunk is temporary disabled for testing purposes (I don't want to download and split dataset every time)

```
# download data
dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)


# name columns
colnames(movies) <- c("movieId", "title", "genres")


# Create Data Frame with movies

# If using R 3.6 or earlier, comment out this statement and use above:
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))

# Join with rating by movieID
movielens <- left_join(ratings, movies, by = "movieId")


# slice of movielens dataset to edx and validation datasets
# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx set
validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set
removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

This chunk is used instead:

```
load("edx.rda")
load("validation.rda")
```

Check datasets dimensions:

Dimensions of Edx dataset are 9000055, 6 and dimensions of validation dataset are 999999, 6

3 Exploratory data analysis

3.1 First look on dataset

```
## [1] "data.table" "data.frame"
```

Class of our dataset is data.frame, we can work with this data class as is.

Let's see on first 6 records in the dataset:

Table 1: Edx dataset first records

userId	movieId	rating	timestamp	title	genres
1	122	5	838985046	Boomerang (1992)	Comedy|Romance
1	185	5	838983525	Net, The (1995)	Action|Crime|Thriller
1	292	5	838983421	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller
1	316	5	838983392	Stargate (1994)	Action|Adventure|Sci-Fi
1	329	5	838983392	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi
1	355	5	838984474	Flintstones, The (1994)	Children|Comedy|Fantasy

We see that the dataset contains records of each rating was done by user and some information about the movie. For example, first line shows that user with ID = 1 had rated movie with ID=122 by five stars. Date and time of the rating can be extracted from the timestamp and we have additional information about the movie such as it's title, combined with year or release and genres of the movie.

The rating is the numeric variable, so it can take any numeric value. But we need to check, if it is a case. Unique ratings we can find in the dataset:

```
##  [1] 5.0 3.0 2.0 4.0 4.5 3.5 1.0 1.5 2.5 0.5
```

Statistics of ratings distribution:

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.500   3.000   4.000   3.512   4.000   5.000
```

Plot of ratings distribution:

If we consider ratings higher than average rating as "positive" and lower than average as "negative", we can find how many positive and negative ratings we have in our dataset:

Table 2: Negative vs. positive ratings

rating_type	n
negative	4494775
postitive	4505280

As we can see, numbers of positive and negative ratings are approximately equal.

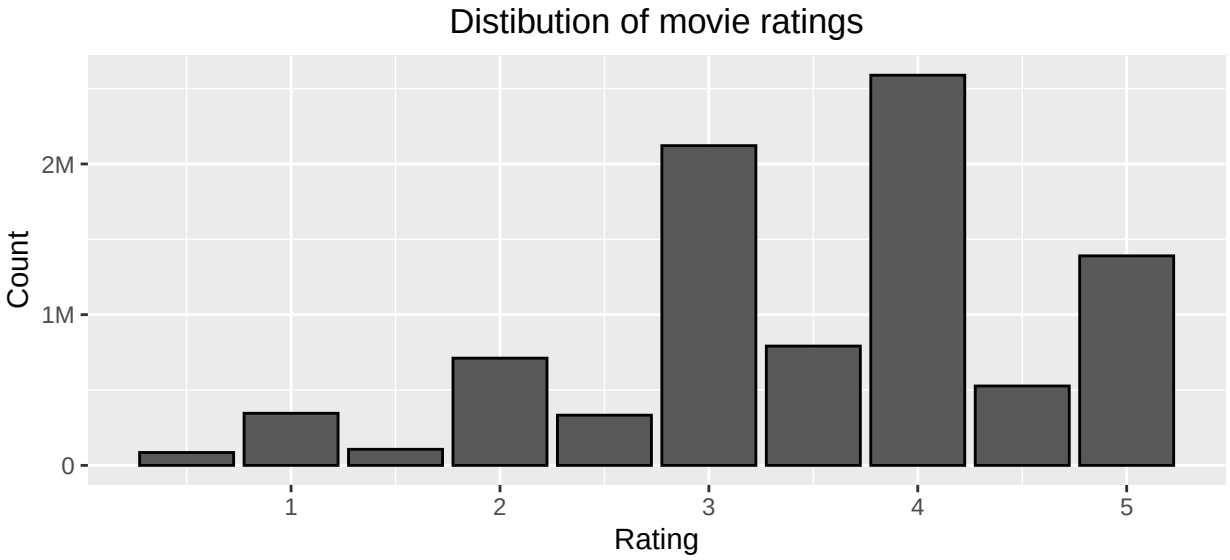

Figure 1: Distibution of movie ratings

Table 3: Half-star vs. whole-star ratings

rating_star	n
half_star	1843170
whole_star	7156885

After first look on the dataset we can conclude:

- Each row in the dataset represents single rating of user defined by userId column to movie, defined by movieId column, additional information about movie (genre and title) and rating timestamp;
- Whole-star rating is much more common than half-star;
- Average rating is about 3.5, but median is 4;
- The most common rating in the dataset is 4, and 50% of all ratings are lying between 3 and 4 (inclusive)

3.2 Movies ans users

Number of unique users in dataset: 69878; unique movies in dataset: 10677.

Total user/movie combinations amount should be: 746087406.

But as we saw before, we have only 9000055 records in the dataset. Only 1.2% of all possible combinations are rated.

To visualize this, we will sample 100 unique users and 100 unique movies and plot matrix with filled cells when user rated the movie and blank cells if not:

Looking on Figure 2 we can rewrite our task: we have to build a model, which can fill the matrix for any given user and any given movie.

3.3 Movies analysis

First, let's look on the top-10 and bottom 10 movies:

Table 4: Best movies by rating

avg_rating	title
5.00	Hellhounds on My Trail (1999)
5.00	Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)

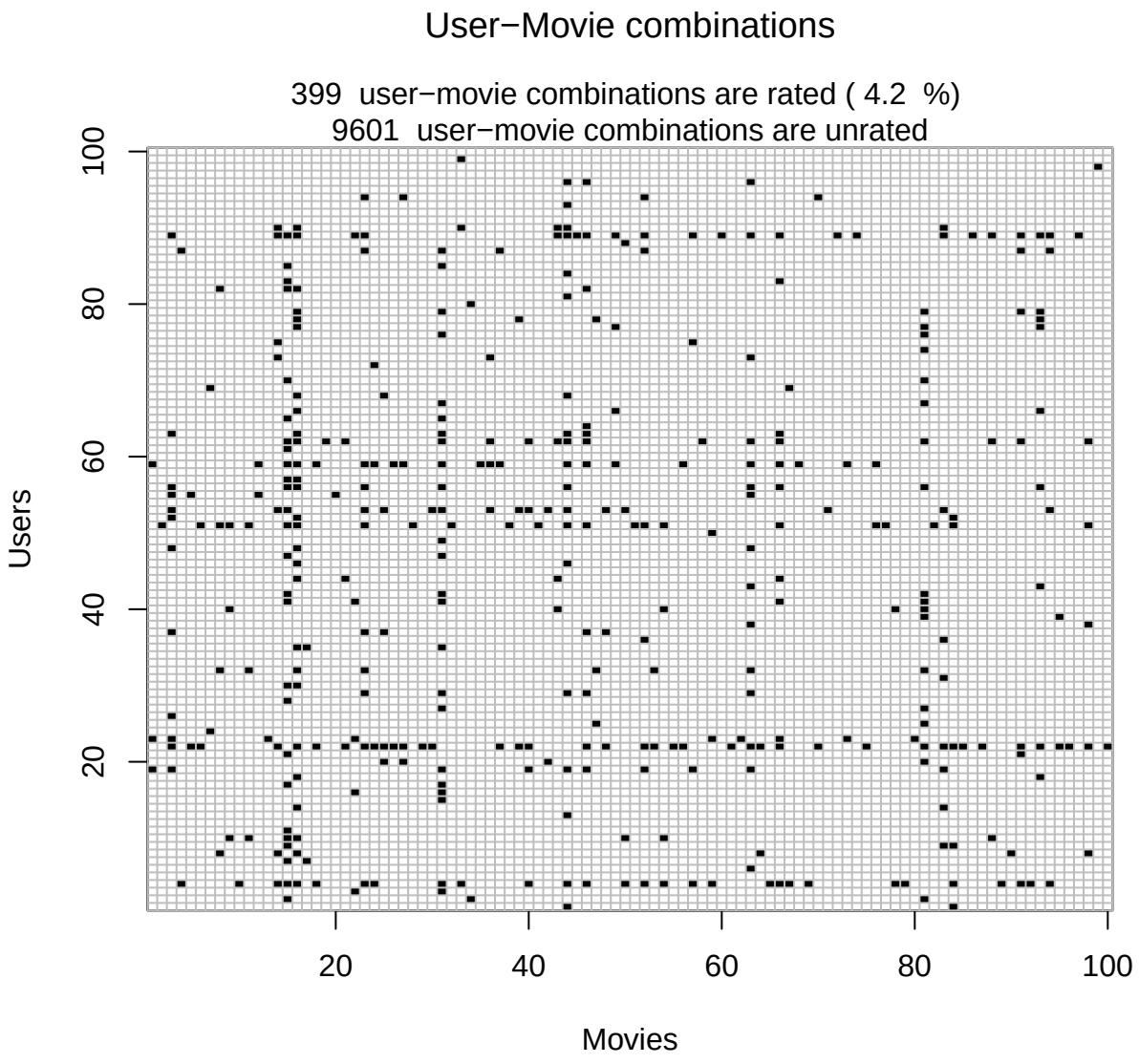

Figure 2: User-Movie combinations

Table 5: Worst movies by rating

avg_rating	title
0.5000000	Besotted (2001)
0.5000000	Hi-Line, The (1999)
0.5000000	Accused (Anklaget) (2005)
0.5000000	Confessions of a Superhero (2007)
0.5000000	War of the Worlds 2: The Next Wave (2008)
0.7946429	SuperBabies: Baby Geniuses 2 (2004)
0.8214286	Hip Hop Witch, Da (2000)
0.8593750	Disaster Movie (2008)
0.9020101	From Justin to Kelly (2003)
1.0000000	Criminals (1996)

Looking on the tables, we see that the top-10 and bottom-10 movies are not widely known by it's title. Let's look on distribution of number of ratings of movies (how many times specific movie was rated):

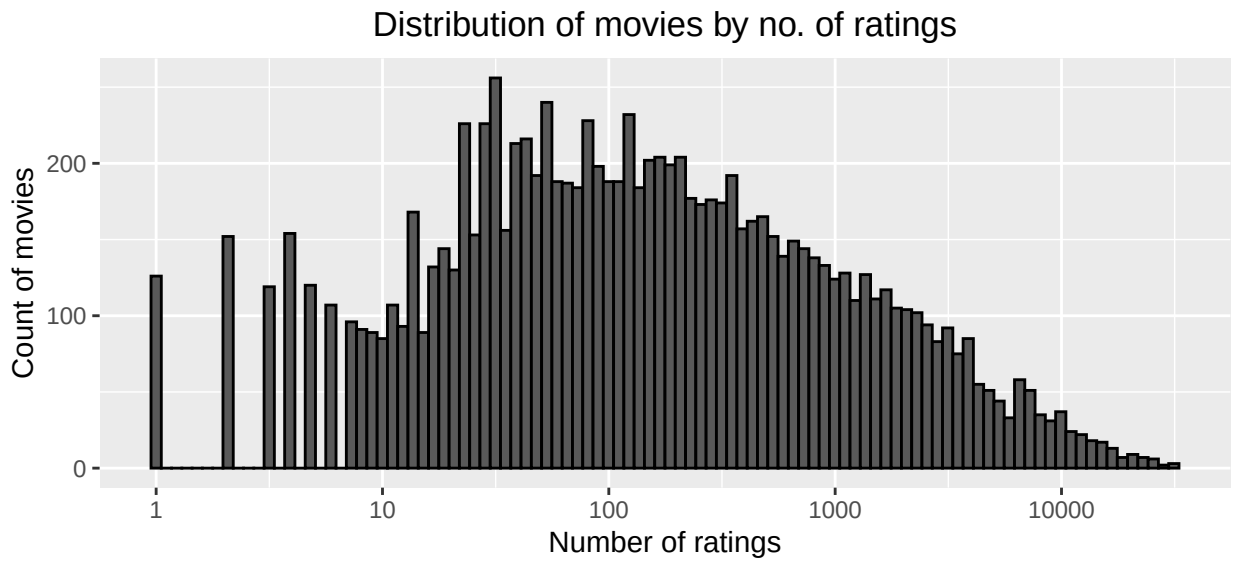

Figure 3: Distribution of movies by number of ratings

We can see, that approximately half of movies have less than 100 ratings and about 125 movies have only one rating. That can explain our observation of top/bottom 10 movies: very high and very low average movie rating can be done based on very few reviews, or even one review. This can't be reliable and will be taken in account when building a prediction model.

To see, how different ratings are distributed across the dataset, we can plot distribution of the average ratings of movies:

Thinking logically, the more ratings specific movie has, the more popular it is. Usually, good movies became very popular, therefore they should have higher average rating. To confirm or reject our hypotheses we can plot average movies ratings versus number of ratings for the movie.

Our hypotheses is confirmed: more often rated movies are also have slightly higher average rating and smaller deviation

Let's look at the top-10 and bottom-10 movies by number of ratings:

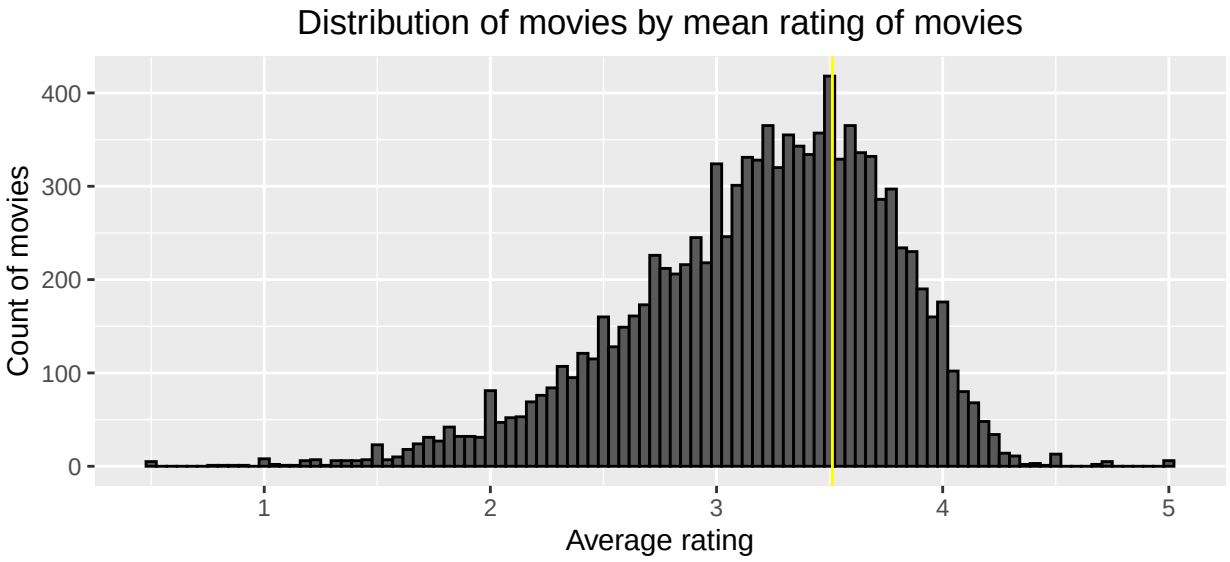

Figure 4: Distribution of movies by mean rating of movies

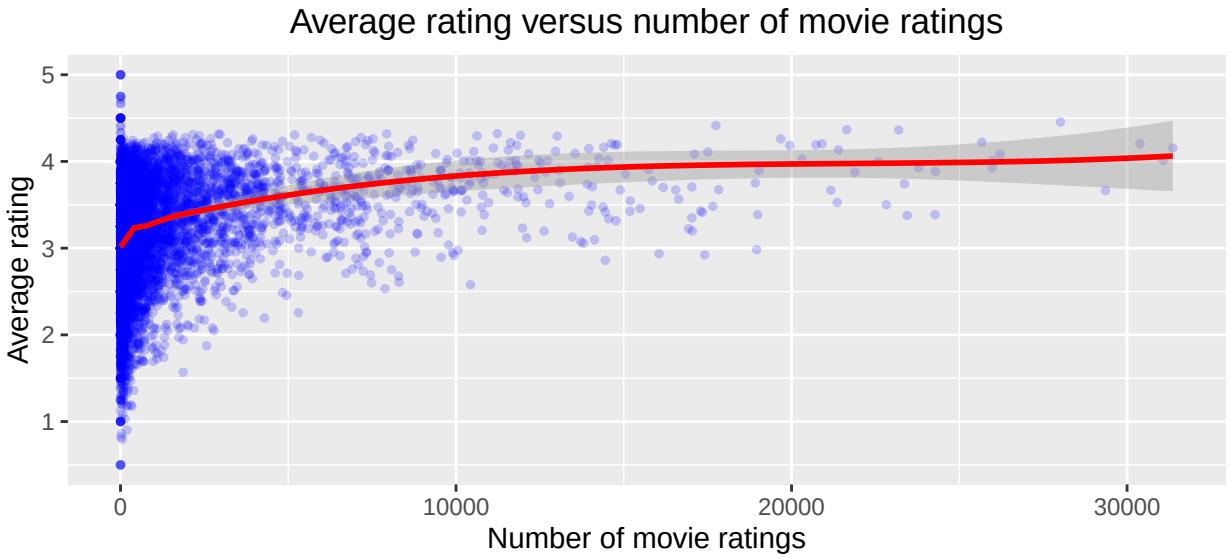

Figure 5: Average rating versus number of movie ratings

Table 6: Most popular movies

movieId	number_of_ratings	avg_rating	title
296	31362	4.154789	Pulp Fiction (1994)
356	31079	4.012822	Forrest Gump (1994)
593	30382	4.204101	Silence of the Lambs, The (1991)
480	29360	3.663522	Jurassic Park (1993)
318	28015	4.455131	Shawshank Redemption, The (1994)
110	26212	4.081852	Braveheart (1995)
457	25998	4.009155	Fugitive, The (1993)
589	25984	3.927859	Terminator 2: Judgment Day (1991)
260	25672	4.221311	Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)
150	24284	3.885789	Apollo 13 (1995)

Table 7: Least popular movies

movieId	number_of_ratings	avg_rating	title
3191	1	3.5	Quarry, The (1998)
3226	1	5.0	Hellhounds on My Trail (1999)
3234	1	3.0	Train Ride to Hollywood (1978)
3356	1	3.0	Condo Painting (2000)
3383	1	3.0	Big Fella (1937)
3561	1	1.0	Stacy's Knights (1982)
3583	1	3.0	Black Tights (1-2-3-4 ou Les Collants noirs) (1960)
4071	1	1.0	Dog Run (1996)
4075	1	1.0	Monkey's Tale, A (Les ChÃ¢teau des singes) (1999)
4820	1	2.0	Won't Anybody Listen? (2000)

Look at these tables confirms, that movies which were rated more often, have higher average rating. We can see that the movie "Hellhounds on My Trail (1999)" also appeared in top rated movies table as it has average rating 5, but it is based only on a single rating, which cannot be reliable. We will take it in account during model building and tuning.

3.4 Users analysis

Now we will perform similar analysis but for users.

4 Literature

1. Francesco Ricci and Lior Rokach and Bracha Shapira, Introduction to Recommender Systems Handbook